Decomposing Utility Functions in Bounded Max-Sum for Distributed Constraint Optimization

Emma Rollon and Javier Larrosa

Universitat Politècnica de Catalunya

Abstract. Bounded Max-Sum is a message-passing algorithm for solving Distributed Constraint Optimization Problems (DCOP) able to compute solutions with a guaranteed approximation ratio. In this paper we show that the introduction of an intermediate step that decomposes functions may significantly improve its accuracy. This is especially relevant in critical applications (e.g. automatic surveillance, disaster response scenarios) where the accuracy of solutions is of vital importance.

Introduction

Bounded Max-Sum (BMS) [11] approximately solves Distributed Constraint Optimization Problems (DCOP) with very little computation and communication demands. Arguably, its most interesting feature is that it comes with a *guarantee approximation ratio*, meaning that its approximate solution has a utility which is no more than a factor away from the optimum. The algorithm has been recently revisited and enhanced, producing two improved versions: IBMS [12] and RN-BMS [9], with tighter upper and lower bounds, respectively.

All the BMS algorithms have a relaxation phase in which some functions are replaced by smaller arity functions. In general, such replacement introduces some error, which prevents the algorithms from computing the true optimal solution. In this paper we study the possibility of an exact decomposition in which those binary functions are replaced by pairs of unary functions which faithfully capture the same information. Since, in the general case, functions do not have an exact decomposition, we consider approximate decompositions in which the error introduced is minimized. We theoretically prove that our approach improves the upper bound obtained by IBMS and show its performance in a set of graph coloring problems.

Preliminaries

In this Section we review the main elements to contextualize our work. Definitions and notation are borrowed almost directly from [11]. We urge the reader to visit that reference for more details and examples.

DCOP

A *Distributed Constraint Optimization Problem* (DCOP) is a tuple $P = (\mathbf{A}, \mathbf{X}, \mathbf{D}, \mathbf{F})$, where $\mathbf{A} = \{\mathcal{A}_1, \ldots, \mathcal{A}_r\}$ is a set of agents, and $\mathbf{X} = \{x_1, \ldots, x_n\}$ and $\mathbf{D} = \{\mathbf{d}_1, \ldots, \mathbf{d}_n\}$ are variables and domains. $\mathbf{F} = \{f_1, \ldots, f_e\}$ is a set of cost functions. The objective function is $F(x) = \sum_{j=1}^{e} f_j(x^j)$ where $x^j \subseteq \mathbf{X}$ is the scope of f_j. A *solution* is a complete assignment $\mathbf{x}$. An *optimal solution* is a complete assignment $\mathbf{x}^*$ such that $\forall \mathbf{x},\ F(\mathbf{x}^*) \geq F(\mathbf{x})$. The usual task of interest is to find $\mathbf{x}^*$ through the coordination of the agents.

In the applications under consideration, the agents search for the optimum via decentralized coordination. We assume that each agent can control only its local variable(s) and has knowledge of, and can directly communicate with, a few neighboring agents. Two agents are neighbors if there is a relationship connecting variables and functions that the agents control.

The structure of a DCOP problem $P = (\mathbf{A}, \mathbf{X}, \mathbf{D}, \mathbf{F})$ can be transformed into a factor graph. A *factor graph* is a bipartite graph having a variable node for each variable $x_i \in \mathbf{X}$, a factor node for each local function $f_j \in \mathbf{F}$, and an edge connecting variable node x_i to factor node f_j if and only if x_i is an argument of f_j.

Bounded Max-Sum Algorithms

The *BMS* algorithm [11] and its improved versions IBMS [12] and RN-BMS [9] are approximation methods built on Max-Sum [4, 1]. From a possibly cyclic problem P, the idea is to remove cycles in its factor graph by ignoring dependencies between functions and variables, producing a new acyclic problem. Then, Max-Sum is used to optimally solve the acyclic problem while simultaneously computing the approximation ratio.

Here, for simplicity purposes, we restrict ourselves to IBMS, which was proven to be always superior to BMS and usually superior to RN-BMS. For the sake of simplicity, we will restrict ourselves to the case of binary functions $f_j(x_i, x_k)$. The extension to general functions is direct.

IBMS works in three phases:

- **Relaxation Phase:** First, the algorithm assigns a weight w_{ij} to each edge (i, j) of the original factor graph. Then, it finds a maximum spanning tree with respect to the weights. Next, the original problem P is transformed into an acyclic one $\widetilde{P}$ having the spanning tree as factor graph as follows: For each edge (i, j) in the original graph that does not belong to the tree, the cost function $f_j(x_i, x_k)$ is transformed into another function $\widetilde{f}_j(x_k) = \max_{x_i}\{f_j(x_i, x_k)\}$.
  Let T denote the set of functions that have not been simplified. The objective function of $\widetilde{P}$ is
  $$\widetilde{F}(x) = \sum_{j \in T} f_j(x_i, x_k) + \sum_{j \notin T} \widetilde{f}_j(x_k)$$
- **Solving Phase:** IBMS solves $\widetilde{P}$ with Max-Sum. Let $\widetilde{\mathbf{x}}$ be the solution given by IBMS. Since the factor graph of $\widetilde{P}$ is acyclic, $\widetilde{\mathbf{x}}$ is optimal for $\widetilde{P}$. IBMS returns $\widetilde{\mathbf{x}}$ as a sub-optimal solution for P.

x_i	x_k	f_j
a	a	15
a	b	30
b	a	10
b	b	25

x_i	g_j
a	10
b	5

x_k	h_j
a	5
b	20

Fig. 1. Example of a binary function f_j that can be exactly decomposed into two unary functions $h_j(x_k)$ and $g_j(x_i)$.

– **Bounding Phase:** IBMS computes a guarantee approximation ratio as follows. Note that $F(\widetilde{\mathbf{x}})$ is an obvious lower bound of the the optimal ($F(\widetilde{\mathbf{x}}) \leq F(\mathbf{x}^*)$). Moreover, it can be shown that $\widetilde{F}(\widetilde{\mathbf{x}})$ is an upper bound of the optimal ($\widetilde{F}(\widetilde{\mathbf{x}}) \geq F(\mathbf{x}^*)$). Therefore, $\rho = \frac{\widetilde{F}(\widetilde{\mathbf{x}})}{F(\widetilde{\mathbf{x}})}$ is a guarantee approximation ratio for IBMS.

Decomposition

The IBMS algorithm relaxes the problem by replacing some binary functions $f_j(x_i, x_k)$ by unary functions $\widetilde{f}_j(x_k)$. Clearly, the relaxed problem is in general not equivalent to the original one because the transformation introduces an error.

Exact Decomposition

The idea of exact decomposition is to replace the binary function $f_j(x_k, x_i)$ by two unary functions $h_j(x_k)$ and $g_j(x_i)$ such that there is no loss of information. Formally, given a binary function $f_j(x_i, x_k)$ we can set a system of linear equations,

$$\forall_{x_i, x_k} \;\; f_j(x_i, x_k) = h_j(x_k) + g_j(x_i)$$

such that $\forall x_k, h_j(x_k) \geq 0$ and $\forall x_i, g_j(x_i) \geq 0$, where each entry of the unary functions is a variable of the system. If the system has a solution, that solution is an *exact decomposition*. Replacing the binary function by the two unary functions modifies the factor graph without introducing any error. The system of linear equations can be solved very efficiently with one of the many Integer Programming toolkits available.

As an example, consider the binary cost function $f_j(x_i, x_k)$, and the two unary functions $h_j(x_k)$ and $g_j(x_i)$ in Figure 1, respectively. Observe that the combination of the two unary functions is equivalent to the binary one. The reason being that the following equations are satisfied,

$$\begin{array}{ll} h_j(a) + g_j(a) = f_j(a, a) & \quad h_j(b) + g_j(a) = f_j(a, b) \\ h_j(a) + g_j(b) = f_j(b, a) & \quad h_j(b) + g_j(b) = f_j(b, b) \end{array}$$

Therefore, in this case, exact decomposition could be achieved.

The natural application of the previous idea constitutes our first algorithm called *exact decomposition-based IBMS* (ED-IBMS). It differs from the previous ones only in

the relaxation phase. Before computing each $\widetilde{f}_j(x_k)$, ED-IBMS attempts an exact decomposition. Let D be the set of functions in which exact decomposition was achieved. The resulting objective function is

$$\widetilde{F}_{ED}(x) = \sum_{j \in T} f_j(x_i, x_k) + \sum_{j \in D} (g_j(x_i) + h_j(x_k)) + \sum_{j \notin T \cup D} \widetilde{f}_j(x_k)$$

It is easy to see that the cost of any solution in the relaxed problem with exact decomposition is smaller than or equal to its cost in the relaxed problem without exact decomposition. In other words, ED-IBMS always obtains upper bounds tighter than IBMS. Formally, $F(x) \leq \widetilde{F}_{ED}(x) \leq \widetilde{F}(x)$ holds.

Approximate Decomposition

Note that exact decompositions do not exist in general. In a preliminary set of experiments we observed that exact decomposition occurs very rarely which makes, in practice, IBMS and ED-IBMS behave identically. When we looked into the details, we observed that very often exact decomposition was *almost* achievable, which leaded us to approximate decomposition.

The idea of approximate decomposition is to replace a given binary function $f_j(x_i, x_k)$ by the combination of two unary functions $h_j(x_k)$ and $g_j(x_i)$, and a binary function $r_j(x_i, x_k)$. Formally,

$$\forall_{x_i, x_k}, f_j(x_i, x_k) = h_j(x_k) + g_j(x_i) + r_j(x_i, x_k) \tag{1}$$

such that $\forall_{x_i, x_k}, r_j(x_i, x_k) \geq 0$, $\forall x_k, h_j(x_k) \geq 0$, and $\forall x_i, g_j(x_i) \geq 0$. As before, this expression can be seen as a system of linear equations where each entry in the unary functions and in the binary function $r(x_i, x_k)$ is a variable of the system. Note that $g_j(x_i)$ and $h_j(x_k)$ represent the part of the utility function $f_j(x_i, x_k)$ that has been decomposed, while $r_j(x_i, x_k)$ represents the part that has not been decomposed (the *residual* utility function).

Moreover, we want to ensure that the decomposition improves the upper bound on the original problem. In other words, the optimum of the relaxed problem with approximate decomposition must be tighter than the optimum of the relaxed problem without approximate decomposition. Formally,

$$\forall_{x_i, x_j} \max_{x_i} \{r(x_i, x_k)\} + g(x_i) + h(x_k) \leq \max_{x_i} \{f(x_i, x_k)\} \tag{2}$$

This inequality can be rewritten using Expression 1 as,

$$\forall_{x_i, x_k} \max_{x_i} \{r(x_i, x_k)\} \leq \max_{x_i} \{f(x_i, x_k)\} - f(x_i, x_k) + r(x_i, x_k)$$

Each inequality can be transformed into a set of inequalities without the $\max$ operator over function $r(x_i, x_j)$ as follows,

$$\forall_{x_i, x_k} \forall_{a \neq x_i}, r(a, x_k) \leq \max_{x_i} \{f(x_i, x_k)\} - f(x_i, x_k) + r(x_i, x_k) \tag{3}$$

As an example, consider function $f_j(x_i, x_k)$ in Figure 2. The system of linear equations required to approximatelly decompose that function is,

x_i	x_k	f_j
a	a	20
a	b	30
b	a	10
b	b	25

x_i	g_j
a	5
b	0

x_k	h_j
a	10
b	25

x_i	x_k	r_j
a	a	5
a	b	0
b	a	0
b	b	0

Fig. 2. Example of a binary function f_j that is approximate decomposed into $g_j(x_i)$, $h_j(x_k)$ and $r_j(x_i, x_k)$.

$$
\begin{array}{ll}
g_j(a) + h_j(a) + r_j(a,a) = 20 & r_j(b,a) \leq \max\{20,10\} - 20 + r_j(a,a) \\
h_j(a) + g_j(b) + r_j(a,b) = 30 & r_j(a,a) \leq \max\{20,10\} - 10 + r_j(b,a) \\
h_j(b) + g_j(a) + r_j(b,a) = 10 & r_j(b,b) \leq \max\{30,25\} - 30 + r_j(a,b) \\
h_j(b) + g_j(b) + r_j(b,b) = 25 & r_j(a,b) \leq \max\{30,25\} - 25 + r_j(b,b)
\end{array}
$$

subject to,

$$
\begin{array}{llll}
h_j(a) \geq 0 & g_j(a) \geq 0 & r_j(a,a) \geq 0 & r_j(b,a) \geq 0 \\
h_j(b) \geq 0 & g_j(b) \geq 0 & r_j(a,b) \geq 0 & r_j(b,b) \geq 0
\end{array}
$$

where $h_j(\cdot)$, $g_j(\cdot)$, and $r_j(\cdot,\cdot)$ are the variables of the system. Figure 2 shows one solution to the previous system.

Any solution to the system of linear equations from Expression 1 and Expression 3 is an approximate decomposition. Some solutions may be better than other. The worst decomposition would be one in which h_j and g_j are zero because no decomposition would have been achieved. In general, one may prefer those decompositions that minimize in one way or another the residuals (note that exact decomposition coincides with zero residuals). We consider two possibilities:

- Minimize the maximum residual: $\min \max_{x_i,x_k} r_j(x_i, x_k)$.
- Minimize the average residual: $\min \sum_{x_i,x_k} r_j(x_i, x_k)$.

Such objective functions can easily be added to the system of equations and subsequently solved with an Integer Programming toolkit.

Approximate decompositions introduce a new family of IBMS algorithms called AD-IBMS. The idea is to compute an approximate decomposition of function $f_j(x_i, x_k)$ before its relaxation. Then, the relaxation is performed not over the original function f_j, but over the residual r_j (i.e., $\widetilde{r}_j(x_k) = \max_{x_i}\{r_j(x_i, x_k)\}$). Thus, the objective function of the relaxed problem is,

$$\widetilde{F}_{AD}(x) = \sum_{(i,j),(k,j)\in T} f_j(x_i, x_k) + \sum_{(i,j)\notin T} (g_j(x_i) + h_j(x_k) + \widetilde{r}_j(x_k))$$

Since the system of linear equations enforce Expression 2, it is easy to see that AD-IBMS always obtains tighter upper bounds than IBMS. Formally, $F(x) \leq \widetilde{F}_{AD}(x) \leq \widetilde{F}(x)$ holds.

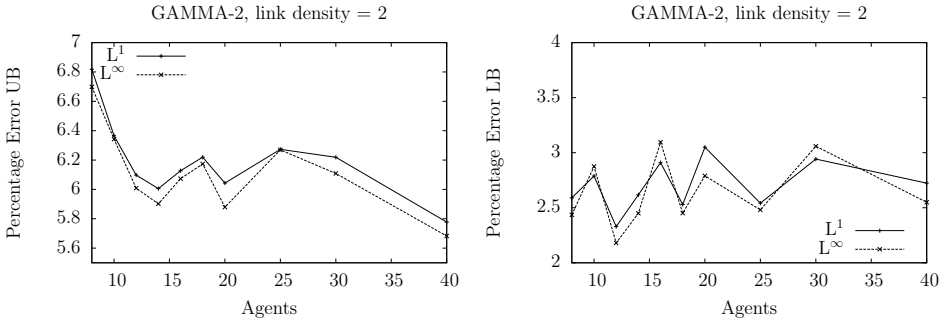

Fig. 3. Percentage error of the upper (left) and lower bound (right) obtained by AD-IBMS minimizing the maximum residual ($L1$) and minimizing the average residual (L^∞).

Empirical Evaluation

The purpose of the experiments is to compare IBMS with respect to AD-IBMS. In particular, we want to evaluate the improvement of the bounds and approximation ratio of the IBMS algorithm using approximate decomposition. For the sake of completeness, we will also report the results for standard BMS and RN-BMS. The percentage error of a value v_{approx} (i.e., upper or lower bound) is computed as $\frac{|v-v_{approx}|}{v} \times 100$ where v is the optimum of the problem.

We consider the same set of problems from the ADOPT repository[1] used in [11]. These problems represent graph coloring problems with two different link densities (i.e., the average connection per agent) and different number of nodes. Each agent controls one node (i.e., variable), with domain $|\mathbf{d}_i| = 3$, and each edge of the graph represents a pairwise constraint between two agents. Each edge is associated with a random payoff matrix, specifying the payoff that both agents will obtain for every possible combination of their variables' assignments. Each entry of the payoff matrix is a real number sampled from two different distributions: a gamma distribution with $\alpha = 9$ and $\beta = 2$, and a uniform distribution with range $(0, 1)$. For each configuration, we report average values over 25 repetitions. For the sake of comparison, we compute the optimal utility with a complete centralized algorithm.

First, we evaluate if the accuracy of AD-IBMS depends on the way the residuals are minimized. Figure 3 shows the percentage relative error of the upper bound (left) and lower bound (right) obtained by AD-IBMS minimizing the maximum residual (L^1) and minimizing the average residual (L^∞). We only report the results on gamma distribution with link density 2 because the behavior pattern is very similar on the other classes of problems. Although theoretically incomparable, L^∞ is always superior to L^1 across all instances. Thus, in the following, AD-IBMS refers to the L^∞ case.

In our second experiment, we compare the bounds obtained by standard BMS, RN-BMS, IBMS, and AD-IBMS. Figure 4 (first two rows) shows the percentage relative

[1] http://teamcore.usc.edu/dcop

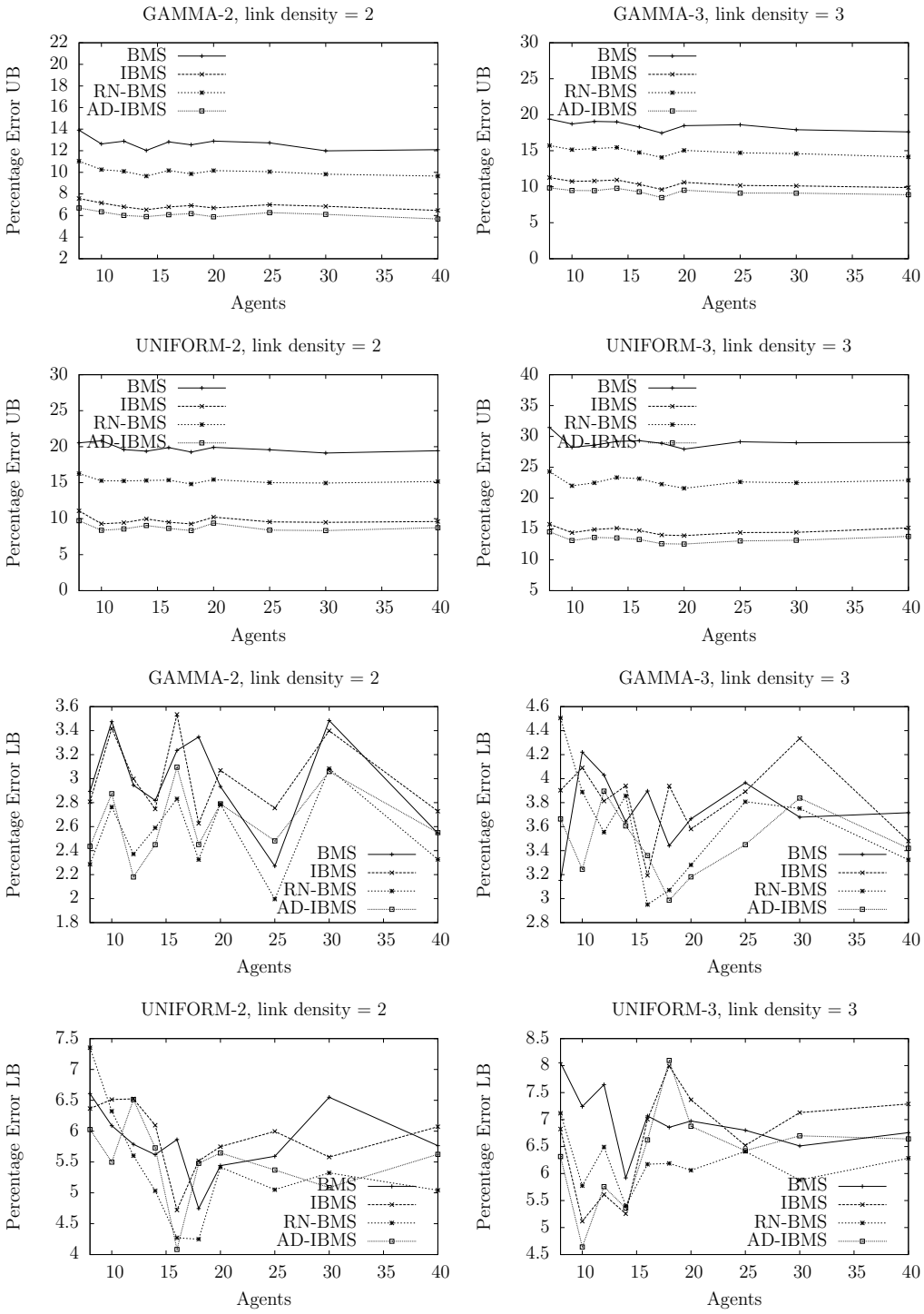

Fig. 4. Percentage error of the upper (first two rows) and lower bound (last two rows) obtained by BMS, IBMS, RN-BMS and AD-IBMS minimizing the maximum residual.

error of the upper bound obtained by each algorithm. The behavior of all algorithms is very similar across all link densities and payoff distributions. As theoretically proved, AD-IBMS always computes the tightest upper bound. Figure 4 (last two rows) shows the percentage relative error of the lower bound obtained by the previous set of algorithms. In general, AD-IBMS obtains more accurate lower bounds than BMS and IBMS. In some cases, AD-IBMS is also superior to RN-BMS. Note that this improvement is very relevant. On the one hand, recall that this class of algorithms are being developed for applications in which the accuracy of the solution is extremely important. On the other hand, since BMS, IBMS and specially RN-BMS are already very accurate on this type of problems, one cannot expect dramatic improvements. This improvement leads AD-IBMS to obtain very tight approximation ratios. Moreover, the computation time of AD-IBMS is always smaller than twice the computation time of IBMS.

The maximum 95% confidence interval for the gamma and uniform payoff distributions is smaller than 5.5 for the upper bound and smaller than 1 for the lower bound. This small confidence interval shows that 25 repetitions provide, for our experimental setting, a good sample size to assess the statistical significance of the results.

Finally, note that in [12] IBMS was shown to be superior to BMS and to two region-optimal criteria introduced in [13] and [14], which were shown to produce tighter approximation ratios than the approach in [6]. Moreover, in [11] BMS was shown to produce tighter approximation ratios than the approach in [2].

Discussion

The idea of exactly decomposing functions into smaller arity ones is far from new. In the field of probabilistic graphical models [7], where functions are (conditional) probability distributions, exact decomposition is a central issue and can be achieved when the probabilistic variables are (conditionally) independent. In the field of constraint satisfaction, where functions are relations, exact decomposition has been studied at least in [10]. The goal there was to compute the minimal network (i.e, transforming large arity relations into sets of equivalent binary ones). More recently, [5] have studied the power of decomposition in the context of combinatorial optimization graphical models. The goal there was to avoid large arity functions in order to boost local consistency enforcement.

Regarding approximate decomposition, the Mini-Bucket Elimination (MBE) algorithm [3] is the closest to ours. MBE is a dynamic-programming approximation algorithm that decomposes large functions into smaller arity ones in order to keep the space complexity manageable. The algorithm is very general and leaves several aspects undefined. [8] proposed an implementation that decomposes the functions while minimizing the error of the decomposition. One of the main differences wrt our approach is the system of linear equations solved.

Acknowledgment

This work was supported by project DAMAS (TIN2013-45732-C4-3-P).

References

1. S. M. Aji and R. J. McEliece. The generalized distributive law. *IEEE Trx. on Information Theory*, 46(2):325–343, 2000.
2. E. Bowring, J. P. Pearce, C. Portway, M. Jain, and M. Tambe. On *k*-optimal distributed constraint optimization algorithms: new bounds and algorithms. In Lin Padgham, David C. Parkes, Jörg P. Müller, and Simon Parsons, editors, *AAMAS (2)*, pages 607–614. IFAAMAS, 2008.
3. R. Dechter and I. Rish. Mini-buckets: A general scheme for bounded inference. *J. of the ACM*, 50(2):107–153, 2003.
4. A. Farinelli, A. Rogers, A. Petcu, and N. R. Jennings. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *AAMAS*, pages 639–646, 2008.
5. A. Favier, S. de Givry, A. Legarra, and T. Schiex. Pairwise decomposition for combinatorial optimization in graphical models. In *IJCAI*, pages 2126–2132, 2011.
6. C. Kiekintveld, Z. Yin, A. Kumar, and M. Tambe. Asynchronous algorithms for approximate distributed constraint optimization with quality bounds. In *AAMAS*, pages 133–140, 2010.
7. D. Koller and N. Friedman. *Probabilistic Graphical Models*. The MIT Press, 2009.
8. D. Larkin. Approximate decomposition: A method for bounding and estimating probabilistic and deterministic queries. In *UAI*, pages 346–353, 2003.
9. J. Larrosa and E. Rollon. Risk-neutral bounded max-sum for distributed constraint optimization. In *SAC*, pages 92–97, 2013.
10. I. Meiri, J. Pearl, and R. Dechter. Tree decomposition with applications to constraint processing. In *AAAI*, pages 10–16, 1990.
11. A. Rogers, A. Farinelli, R. Stranders, and N. R. Jennings. Bounded approximate decentralised coordination via the max-sum algorithm. *Artif. Intell.*, 175(2):730–759, 2011.
12. E. Rollon and J. Larrosa. Improved bounded max-sum for distributed constraint optimization. In *CP*, pages 624–632, 2012.
13. M. Vinyals, E. Shieh, J. Cerquides, J. A. Rodriguez-Aguilar, Z. Yin, M. Tambe, and E. Bowring. Quality guarantees for region optimal dcop algorithms. In *AAMAS*, pages 133–140, 2011.
14. M. Vinyals, E. Shieh, J. Cerquides, J. A. Rodriguez-Aguilar, Z. Yin, M. Tambe, and E. Bowring. Reward-based region optimal quality guarantees. In *OPTMAS Workshop*, 2011.